Andrew Chabot
CSCI-711

Global Illumination Project Report: Cutting Ice

Introduction

With this project, I set out to extend but more so use the ray tracer to be able to make a video simulating cuts on ice. Originally, I was unsure of whether to do this project in Unreal or add it to my ray tracer engine. I chose to use my ray tracer mostly for fun, as a more homebrewed solution seemed like it could be an enjoyable project to work on. The whole idea here is to create a somewhat immersive experience with snow falling down onto ice that is being cut. This idea originated from hockey skates and the blade of a skate cutting through ice. The project is capable of making these cuts but the main video I generated for this project is a bit different in scope than the original plan. Instead of cuts being made from user input randomly on the ice, a large block is cut out of the ice and then raised. This was done to show some of the transmissive properties of ice and for increased visual appeal. The documentation for running this project is explained lightly in the system section but can also be seen in the README.md file in the submitted code zip file.

Architecture

The main code for this system is in the ice_sim.py file. This file contains the functions for cutting ice, making videos from images, and controlling the overall World that is being rendered for each frame. Starting with the ice, this is a slightly blue, highly transmissive polygon. The required packages for running this are numpy, opencv-python, multiprocessing, math, and matplotlib. The ice stretches 100x100 pixels and 1 pixel deep. There is a large, red, reflective sphere under the ice to help show the properties of the scene with cuts being made in the ice and snow falling down.

The snow is implemented as a child class of the sphere class in my ray tracer program. These are very reflective and white spheres. The size of the snow particles varies but they tend to have very small radiuses. The x,y, and z coordinates are all varied within a certain range as well. The most important attribute of the snow is the falling() function that they have. This falling function enables the snow to move towards the ground frame-by-frame. Each snow object is given a falling value at initialization that will be constant through the entire video. The video is created through opencv-python's VideoWriter functionality. The videos that are created are all .avi format. Each frame of the simulation is ray-traced and added onto the video iteratively.

It's important to note that I am running the ray tracing in parallel using multiprocessing Pools. I render rows in parallel and combine them back together after the Pools have finished. This is almost fully parallel because of the way I do it and speeds up the rendering by a factor of around 5 on average.

System

The system takes in a few different parameters from the user for the different modes that it can run in. The first is the number of cuts to make on the ice. There are two main "modes" for doing this in. The first mode makes a certain amount of cuts immediately in the ice and lets the snow fall. This mode is run through any positive input in this field, including 0. The second mode cuts a block in the ice and raises this block up the screen. This mode is run through entering a -1 in this field. The next field is somewhat self-explanatory and is for the number of snow particles to use in the scene. These will be taxing on performance, so I wouldn't recommend over 100 in most circumstances. The next parameter is the number of frames to render for the video. If you wish to create an image instead of a video, you can put 0 in this field. Finally, the last user parameter is the name of the image or video needs to save to. The extensions here are important, for a video the extension must be .avi and for an image .png is a safe extension.

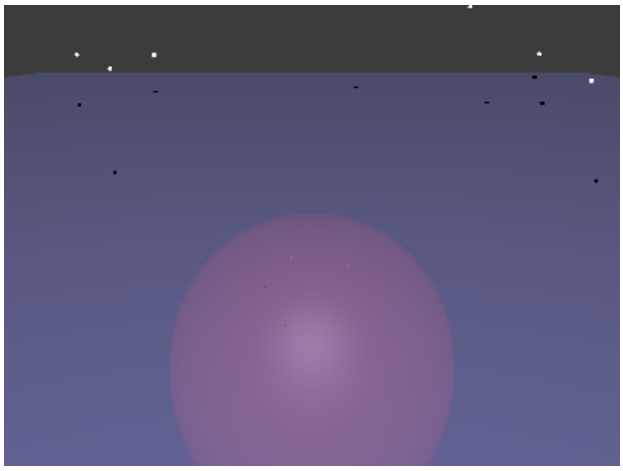
How the system actually works is by treating a cut in the ice as the creation of a hole in the ice. The ice sheet is then built around this hole in four directions going from the sides of the hole to the edge of the ice sheet. When multiple cuts are made, this process is changed so that the cuts are sorted by x value location and the ice sheet is built from one side of the ice to the first cut, between each cut, and from the last cut to the far edge of the ice sheet. This process is done in both the cutting mode #1 and cutting mode #2. The second cutting mode came about mainly because I wasn't impressed by the first mode at all. The cuts just didn't look realistic and there was nothing that interesting going on with any of the generated scenes. The extraction of a block of ice makes for a cooler effect in my opinion. If snow falls on that block it stops where the block is being raised and will be raised with the block in subsequent frames.

The second cutting mode also shows the actual cuts being made around the ice block before "extraction" of the block. Each side will be cut, then change direction and cut, until the ice block is completely cut out. This just means that the cut will be made in a semi-random rectangle in an area to the front and right of the ice sheet. The objects that show the cut happening are added and removed for each frame of the ray tracer, which isn't efficient. However, efficiency for something like this doesn't really matter when compared to the efficiency of the actual ray tracing parts.

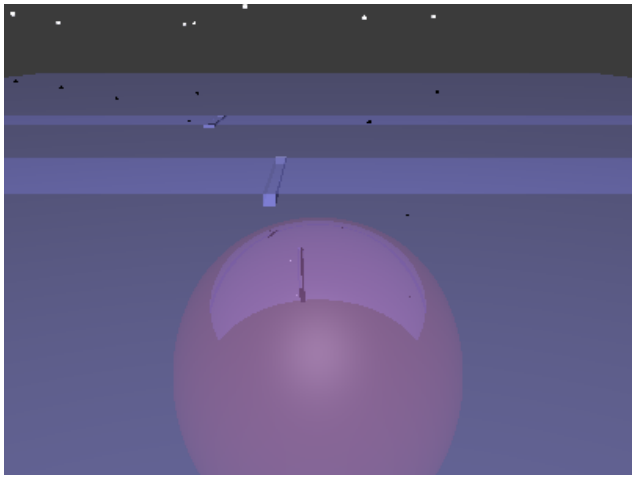
Results

The overall results for this project(especially for mode 2) are best seen in video format, as can be seen on my website at https://andrew-chabot.com/global-illumination-project/ but some still images are shown below. I was able to generate a 250 frame video with cut mode #2 with 100 snow particles. I rendered at 10 frames per second as I found this was a good medium between efficiency in length of video and not looking too choppy. This took me about 6 hours to successfully render at a

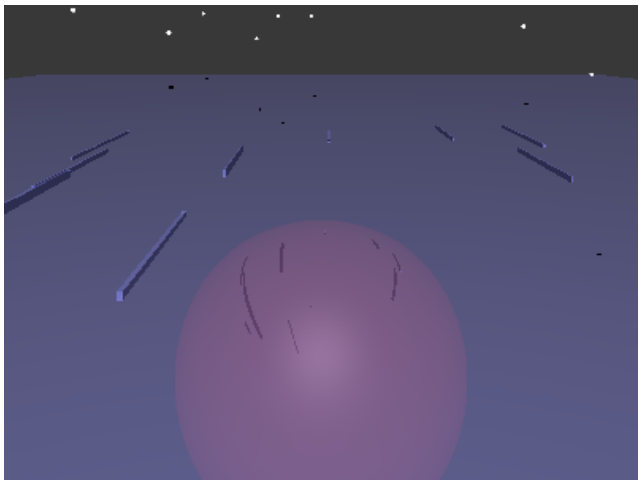
300x400 resolution on a 12 core processor. There is also a short video attached in the code called “short-vid.avi” in case the website videos aren’t working.



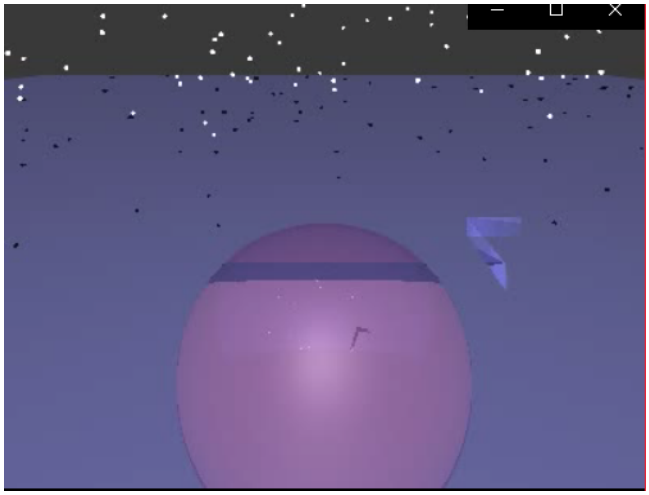
This image has no cuts in the ice and is the base image



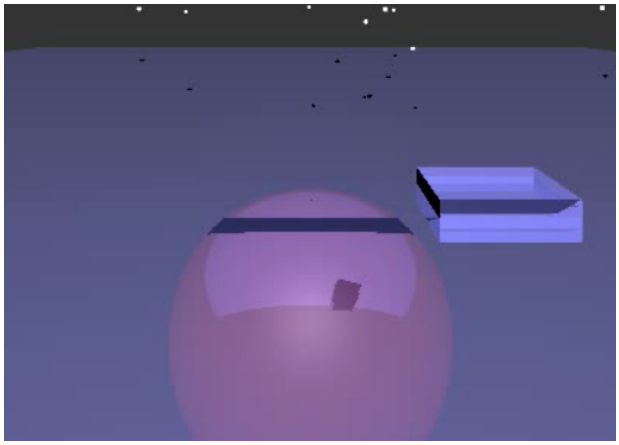
This is an earlier version of the project with two cuts. There was an issue with sheets of ice being drawn in the same place twice and that can be seen here with the horizontal lighter lines matching up with the cut lines.



This is the most updated version of cutting the ice in mode #1. The cuts can be seen as long and skinny holes essentially in the ice. There are 10 present in this image.



Here is an earlier still frame taken from a video while the ice block is being cut in the ice. This is an example of cutting mode #2.



Here is another image of mode #2 with the ice actually being raised out of the area it was cut from. This is a different render than the above image with less snow objects.

Future Work

The main piece of future work to be used would be to use Kd-Trees instead of a simple object list. This would greatly improve the efficiency at which scenes are rendered. The reason I didn't do this was mostly time based. Kd-Trees were attempted, but my attempt did not go well and I decided to explore other advanced checkpoints instead. This would've been the single best way to improve the overall ray-tracing efficiency since there are so many objects in each scene. Other things could be added to increase the atmosphere of the videos created. Fog would be nice here and would help the snow feel more like snow and less like white spheres. I attempted a very basic fog estimation by looking at distance and modifying color based on distance, but this didn't work nearly as well as I desired, since most of the time I was getting infinite distance because of the transmission of ice reaching no objects eventually.

The other big extension would be allowing for horizontal cuts as well as vertical cuts (or diagonal cuts. Currently in the cut mode 1, only vertical cuts are made, which doesn't really look realistic. This opens up a big hole in terms of redrawing the ice around the cuts, which is something I had trouble with even for just the vertical cuts.

Conclusion

The presented system is able to effectively simulate a block of ice being cut out of a sheet of ice and raised out of the sheet. This is more of a cool effect than any sort of real simulation, but the resulting video is at least interesting. There are real efficiency problems here but the resolution isn't terrible for the amount of frames generated. The framerate could certainly be increased, but it could be worse. Using Python was not ideal in terms of efficiency, but it got the job done in the end. This was a fun project to work on and I wish the cutting was more realistic than it ended up being, but the overall result is something that is still cool to show off.